Future of User Storage at CERN

CERN Future of Storage Working Group
X. Espinal, J. Moscicki, A. Peters, D. Van der Ster, A. Wiebalck

First draft 18 March 2020
Revision-v1 15 Octobre 2020/Revision-v2 1 November 2020/Revision-v3 1 December 2020

1 Storage Working Group Mandate

The scope of the working group is to provide a multi-year vision about solutions to provide user-oriented, i.e. non-physics, storage for the CERN community, in particular for use-cases on Linux PCs, lxplus, and lxbatch. To start elaborating a vision the working group decided to focus on the following topics:

- Definition: offer a technical definition of the term *home directory* to avoid confusion. Right now the term is applied in different systems with different meanings: AFS, DFS, CERNBox and Laptop/Desktop.
- Identify the use-cases for user-oriented storage: Separate convenience from requirement, prioritise the features.
- Identify the challenges with the current AFS usage. Clarify the concrete challenges and showstoppers. These could be used to help guide future technical solutions.
- Technology Survey: review the options available in relation to the above use-cases, and estimate their initial and long-term costs.
- Propose one or more future scenarios: combining one or more of the above technologies, present different scenarios, which may vary by feature completeness, initial and long term costs.

2 Identify the challenges with the current AFS usage

The predominant accesses to AFS come from desktop machines, lxplus and batch worker nodes, each with approximately 1/3 of the share respectively. The total average access rate is 10kHz which induces an average load of 300Hz per AFS server.

In terms of space, user home directories are 10GB on average and they are mainly used for interactive work. Users log in to find a consistent environment where they run the common tools for software development. The total size of these home directories is 50TB.

About 50% of the space and 10% of the files in AFS workspaces are occupied by ROOT files that should be stored from elsewhere (EOS). The total amount of space for AFS workspaces is 360TB. Three AFS projects occupy 50% of the total size: ATLAS, SW and Indico, totalling 140TB. We started to report the mis-use of AFS workspace, used to store ROOT files, to the experiments and invite them to revise/clean-up.

The original motivations for the AFS phase-out have evolved. The community is still alive, i.e. companies like Sine Nomine and Auristor offer options for AFS support. In the past years, OpenAFS has regularly published new releases. Recently the IPv6 is on the roadmap for release production v2.0 (current development is v1.9). Still some items should be properly followed-up: authentication schemes is at the moment limited to kerberos, and majority of the current production workflows rely on AFS. From the TCO perspective and given the criticality of the service, the effort needed for the AFS Service is relatively low: around 0.5 FTE. At least the manpower costs need to be reviewed should we plan to keep AFS running for longer time, as the service has been moved to a maintenance mode during the last years.

Conclusion: The size and the access rate make replacing AFS seem deceptively simple. The biggest challenges we identified are two: 1) the current production usage across CERN communities and 2) the scalability need in number of clients.

3 lxbatch use-case of Linux home directories

Around 40% of the batch workload is submitted via local condor submit. With the remainder coming from grid workflows where there is no need for a shared file system. Although to ensure a good job mix and absorb significant variations in the relative fraction, most nodes accept both grid and local jobs.
The most common workflow our users are following for local job submission is to log (ssh) to lxplus, cd into the condor submit dir and run condor submit command, which relies on a shared filesystem (needed by the scheduler). This requirement is not strictly necessary from the technical point of view: HTCondor at CERN additionally supports staging files in and out of the workers using the scheduler, though the interaction pattern with the system is different. This mechanism is in production use by some users and experiments, but most have preferred to stay with the shared filesystem mechanism to avoid having to change their production workflow systems, originally developed for the LSF-based service. In the context of batch submission, the home directories are mounted on lxplus and the batch nodes: Around 100 lxplus nodes are used as remote submit machines where users prepare their code and jobs. There are around 20 HTCondor schedd servers which maintain responsibility for the jobs after they have been submitted and handle the file copying. These access AFS for the sandbox files, such as the job parameter files and the job output. Around 20k batch worker nodes mount the home directories so that the the jobs can access files directly. Non-grid use cases expect access to a POSIX file system from their batch jobs. Furthermore, there is an implicit "contract" with our users to ensure that lxbatch looks the same as lxplus. This helps with the development and testing of batch jobs. Users appreciate the short turnaround time with local submission provided by the current model and any changes to this would need to take this contract into account.

Besides home directories, project areas are also often used for production activities and providing the experiment's required frameworks.

Conclusions: The simplicity of the current lxplus/lxbatch shared-filesystem model is well-appreciated by users, though other models are supported and are in use, e.g. not using a shared filesystem and staging the files in and out via the schedd. The cost of adapting user and experiment production frameworks to any other model is significant and is entirely on the user side. At present the shared filesystem access for job input and output continues to be presented as the easiest option to the users.

4 lxplus use-case of Linux home directories

Lxplus is the central IT supported/managed Linux computing environment. We briefly discussed alternative models: personal VMs, containers instantiated on demand, and notebooks. These alternatives have varied resource costs, feature capabilities, and implications for the users.

Conclusions: While never seriously explored, alternatives to the current lxplus model would come with different pros and cons, and it is unclear if such alternatives would bring a benefit to our user communities. It is very likely that several alternatives would need to be provided to fulfill all user requirements, similar to what is done at the moment.

5 Technology survey

This technology overview is based on some of the concepts that describe the ability of a shared file system to perform: in terms of security, permissions, performance, etc. These different concepts are summarised on a table in section 5.2.11 where we compare some of the widely used available technologies in the market with large scale deployments. These different technologies are: NFS-appliances, CephFS, AFS, Lustre and EOS.

5.1 Security and Permission Models

Security and trust models differ substantially for each implementation. We distinguish a client and a user based authentication model. CephFS provides client based authentication, others support in various flavours client and user based authentication. Client authentication typically uses a shared secret. The authentication model is coupled to authorization models. The main difference in authorization models is if access control is done server side, client delegated or client enforced. The CephFS model provides each client with a global pool access key, filesystem access control is afterwards only applied client-side. EOS uses client delegation and server side enforcement. Others are similar in this respect, e.g. Lustre supports light-weight shared secret authentication and strong Kerberos 5 user authentication.

Dynamic (virtual) group support is important with large user bases to express permissions for certain individuals in the institution. NFS, CephFS and Lustre rely on UNIX mechanisms to express groups. This mechanism is not optimized for rapid changing and large user groups. AFS allows to configure dynamic s.c. AFS groups using the protection server commands (PTS). EOS supports CERN's e-group model, where only the membership in a group has to be verified. Entire groups, which can have more than 10.000 members, don't have to be cached.

5.2 Features, Performance & Scalability

5.2.1 High Availability

The high-availability models in the listed file system implementations are different. Lustre, Netapp and AFS delegate most of the high-availability features for data and meta-data to the hardware itself using hardware redundancy, virtual IPs etc. CephFS implements meta-data high-availability using standby nodes (PAXOS for the central cluster metadata). EOS uses a similar approach, however automatic failover is not used in production.

High availability for data is implemented in EOS and CephFS using replication and erasure encoding.

5.2.2 Scalability

Scalability is a key characteristic for distributed file systems. It can be divided into several areas:

- Data volume scaling
- Metadata volume scaling
- Metadata operation scaling

which are either following a scale-out or an scale-up philosophy. Lustre, CephFS, EOS allow vertical volume scaling (scale-out). In NFS and AFS the volume scaling is coupled to the namespace structure e.g. volumes are attached into the logical namespace e.g. in general this represents more a scale-up architecture. Only CephFS implements metadata volume scaling with a scale-out architecture (RADOS) but due to the limited size of meta-data and the hierarchical nature this is less of a concern.

Again due to the hierarchical nature there is no solution which can scale linear the metadata operation performance - a hierarchical namespace can not be sharded. All implementations allow to scale the meta-data operation performance using subtree branching. CephFS is the only implementation where this can be done dynamic e.g. within a few seconds parts of the namespace can be moved between metadata servers to balance performance.

5.2.3 Backups

None of the solutions, except AFS, provide an internal implementation of backup. Many standard tools allow to backup filesystems, but there are also specialized open-source solutions like restic, which can backup the POSIX attributes of filesystems. AFS is special within this respect due to the very non-POSIX group and permission model.

5.2.4 Snapshots

Snapshots are point-in-time backups of a filesystem, which provide a consistent picture of the whole system at a given moment. NFSapp, CephFS and AFS provide this feature which is very useful in particular for backups.

5.2.5 Versioning

None of the filesystem provides versioning as VMS did in the past. EOS supports versioning with time-based purging. This is however not directly triggered using the filesystem interface. Snapshots in CephFS could provide something similar to versioning. Manual versioning is possible in all filesystems.

5.2.6 Quota

We can distinguish the following models when implementing resource limitations:

- user quota - volume/inodes per user in the whole filesystem
- group quota - volume/inodes per group in the whole filesystem
- user tree quota - volume/inodes per user in a given subtree
- group tree quota - volume/inodes per group in a given subtree
- project quota - volume/inodes for all users in a given subtree
- volume quota - volume/inodes for a user in a volume (AFS)
- pool quota - volume for a given user in a pool (CephFS)

All implementations provide a type of quota enforcement. Also here the implementation, where these are enforced, differ. CephFS has the least flexible quota model providing quota in a subtree, which is enforced client side. EOS has the most flexible quota mode, because it allows to define user, group and project quotas in arbitrary subtrees. While not inherent to AFS, there is external tooling used heavily at CERN which provides a similar flexibility for quota management. NSFapp is the only implementation providing nested tree quotas.

5.2.7 Future Sustainability

All implementations are/will be supported for CentOS 8. Currently there are no native Windows or OSX clients for CephFS, but Windows is in development. The recommended solution for CephFS is NFS access via Ganesha gateways or Samba for all non-Linux platforms. EOS does not provide a native client for Windows. The recommended access is using Samba gateways. OpenAFS is available on all platforms, though with weaker support on Windows and MacOS platforms, so it does not require gateway setups, same for NFSapp. Lustre is a Linux only product.

5.2.8 Licenses

Besides NFSapp and the Auristor commercial version of AFS, all other filesystems are open-source and available under such licenses.

5.2.9 Support

NFSapp and Lustre are backed by companies for support. Commercial support for AFS and Ceph is available as well.

5.2.10 Filesystem Client

OpenAFS, NFSapp, CephFS and Lustre provide a native kernel client. CephFS provides additionally a FUSE client, EOS provides only a FUSE client.

FUSE is in general at least a factor 2-4 slower than a kernel client for meta-data operations. Without splicing a factor 5-10 slower for data transfers. However this limitation do not always matter due to limitations when communicating to metadata and data servers.

5.2.11 Client Maturity, Scalability & Stability

AFS is most mature and stable client implementation (according to our experience). Lustre is proven to be the most scalable implementation concerning the number of filesystem clients in SCs. AFS is assumed to scale better than CephFS, EOS and NFSapp with respect to this e.g. CERN is running over 30k AFS clients in the data centre.

	NFSapp	**CephFS**	**AFS**	**EOS**	**Lustre**
Security					
Kerberos	Yes	No	Yes	Yes	Yes
Shared Secret	Yes	Yes	Yes	Yes	Yes
Static keys	No	Yes	No	No	No
Permissions					
ACL model	nfsv4	posix	afs	rich/eos	posix
Nested groups support	unix	unix	afs	unix/e-groups	unix
Enforcement	server	client	server	server	server
Features/Performance					
metadata HA	hw	hw+sw	hw	sw/man	hw
data HA	hw	hw+sw	hw	hw+sw	hw
Scalability: data volume	scale-up	scale-out	scale-up	scale-out	scale-out
Scalability: metadata storage	scale-up	scale-out	scale-up	scale-up	scale-out
Scalability: subtree branching	static	dynamic	static	static	static
backup	external	external	external	external	external
Snapshots	Yes	Yes	Yes	Devel	Propsal
Versioning	No	No	No	Yes	No
Quota model	u/g/t	t/p	vol	ut/gt/t	u/g/t
Nested Tree Quota	Yes	No	No	No	No
Quota enforcement	server	client/server	server	server	server
Future sustainability					
OS support avail	CentOS 8	CentOS 8	CentOS 8	CentOS 8	CentOS 8
UDP/IPv4	No	No	Yes	No	No
IPv6	Yes	Yes	No	Yes	Yes
Infiniband	Yes	Untested	No	Proposal	Yes
License	Closed	LGPL	IBM Public	GPL3	X GPL2
Client implementation	mount(k)	mount(k+f)	mount(k)	mount(f)	mount(k)
Client Scalability	O(3)	O(3)	O(4)	O(3-4)	O(5)

6 Conclusions

The working group did not identify an urgent need to change the current home directory approach based on AFS. The AFS project and the AFS community recovered from a risky period where the future was not clear and various concerns were raised. The past few years have seen regular OpenAFS releases and there is an AFS client on CentOS 8 (End of Life of CentOS 8 is in 2029). At the same time AFS is tightly integrated with many production workflows, and there are no imminent requirements or issues which are not addressed. Hence, CERN IT should continue to support AFS at an appropriate level.

However, the working group also agreed on the current CERN IT strategy to reduce the dependency on and utilization of AFS and to confine its usage mainly to shared file systems needs (lxbatch and lxplus). The ongoing work in this area should continue. Potential evolution of the current model assumed by central Linux services should also be considered.

The working group sees an interest in moving out of AFS as a long term plan, mainly because of software aging issues that have been exposed in the technology survey, such as the support for IPv6, HA, etc. In addition, the current production use of AFS and the AFS dependencies in our communities render the community effort for a potential migration quite substantial, entailing a long turnover time of at least several months. For this reason reducing the usage of AFS is crucial and should be accompanied by the gradual phase-in of a future replacement.

The working group hence recommends to plan for a longer term activity to look into an AFS replacement technology including all concerned stakeholders.

The approach should be revisited later on, maybe at the end of Run-3.